We claim:

1. In a computer system operating in compliance with Real Time Specification for Java (RTSJ), a method of sharing memory resources between a plurality of JVM processes comprising the steps of:

a) allocating a memory area to store class data by a first JVM process;

b) storing class data in said memory area by said first JVM process;

c) determining if said memory area is allocated by a second JVM process;

d) accessing and utilizing said class data from said memory area by said second JVM process.

2. A method as set forth in claim 1, wherein the step of allocating a memory area to store class data by a first JVM process further comprises the step of setting a semaphore to notify subsequent JVM processes of the existence of a class memory area.

3. A method as set forth in claim 1, wherein the step of accessing and utilizing said class data from said memory area by said second JVM process further comprises the steps of:

a) identifying the location of the class memory area by said second JVM process;

b) said second JVM process communicating with the first JVM process via IPC to reference the class data stored in the class memory area.

4. A method as set forth in claim 1, further comprising the steps of:
separating the class data into application classes and system classes;
storing the system class data in the said memory area.

5. The method as set forth in claim 1, wherein a reference counter is used to determine when all JVM processes have exited.

6. The method as set forth in claim 1, wherein said at least one second JVM process accesses said memory area by mapping the said memory area to the second JVM process using pointers.

7. A computer program product for sharing memory resources between a plurality of JVM processes operating in compliance with the Real Time Specification for Java (RTSJ), comprising computer executable instructions for:
allocating a memory area to store class data by a first JVM process; and
accessing said memory area by at least one second JVM process.

8. A computer program product as set forth in claim 7, further comprising computer executable instructions for:
separating the Java class data into application classes and system classes; and
storing the system class data in the said memory area.

9. A computer program product as set forth in claim 7, further comprising computer executable instructions for identifying the existence of a system class memory area by using a semaphore.

10. A computer program product as set forth in claim 7, further comprising computer executable instructions for determining when all JVM processes have exited.

11. In a computer system, a method for sharing memory resources between a plurality of JVM processes comprising the steps of:
separating class data into system class data and application class data;
allocating a system class memory area to store system class data by a first JVM process;
storing said system class data in said system class memory area;
identifying said system class memory area by a second JVM process; and
accessing and utilizing said system class data from said system class memory area by said second JVM process.

12. A method as set forth in claim 11, wherein the step of allocating a system class memory area to store system class data by a first JVM process further comprises the step of setting a semaphore to notify subsequent JVM processes of the existence of a system class memory area.

13. A method as set forth in claim 11, wherein the step of accessing and utilizing said system class data from said system class memory area by said second JVM process

further comprises at least the steps of:
identifying the location of the system class memory area by said second JVM process; and
said second JVM process communicating with the first JVM process via IPC to reference the system class data stored in the system class memory area.

14. A method as set forth in claim 11, further comprising the steps of:
determining when all JVM processes have exited; and
deallocating said system class memory area.

15. A method as set forth in claim 11, wherein the step of accessing and utilizing said system class data from said system class memory area further comprises at least the step of mapping said system class memory area to said second JVM process by using pointers.

16. A computer program product for sharing memory resources between a plurality of JVM processes, comprising computer executable instructions for:
separating class data into system class data and application class data;
allocating a system class memory area to store system class data by a first JVM process; and
accessing said system class memory area by a second JVM process.

17. A computer program product as set forth in claim 16, further comprising computer executable instructions for identifying the existence of a system class memory area by using a semaphore.

18. A computer program product as set forth in claim 16, further comprising computer executable instructions for:
determining when all JVM processes have exited; and
deallocating said system class memory area.

19. A computer program product as set forth in claim 16, wherein the computer executable instructions for accessing said system class memory area by a second JVM process further comprise instructions for mapping the system class memory area to the second JVM process through the use of pointers.

20. A system for sharing memory resources between a plurality of JVM processes operating in compliance with the Real Time Specification for Java (RSTJ), comprising;
means for allocating a class memory area to store class data by a first JVM process; and
means for accessing said class memory area by a second JVM process.

21. A system as set forth in claim 20, further comprising:
means for separating the Java class data into application classes and system classes; and
means for storing the system class data in the said memory area.

22. A system as set forth in claim 20, further comprising a means for identifying the existence of a system class memory area through the use of a semaphore.

23. A system as set forth in claim 20, further comprising a means for determining when all of the JVM processes have exited.

24. A system as set forth in claim 20, wherein said second JVM process has a shorter start-up time than said first JVM process.

25. A system for sharing memory resources between a plurality of JVM processes, comprising;
means for separating class data into system class data and application class data;
means for allocating a class memory area to store class data by a first JVM process; and
means for accessing said class memory area by a second JVM process.

26. A system as set forth in claim 25, further comprising a means for identifying the existence of a system class memory area through the use of a semaphore.

27. A system as set forth in claim 25, further comprising a means for determining when all of the JVM processes have exited.

28. A system as set forth in claim 25, wherein said second JVM process has a shorter start-up time than said first JVM process.